# Novel Applications of Image-Processing Techniques to Particle Physics

Undergraduate Research Thesis

Presented in Partial Fulfillment of the Requirements for Graduation with Honors Research Distinction in Physics in the Undergraduate Colleges of The Ohio State University

by

Garrett Merz

The Ohio State University

May 2016

Project Advisor: Professor Richard Hughes, Department of Physics

## Abstract

Perhaps the most important unknown property of the newly-discovered Higgs boson is how strongly it couples to the top quark, the heaviest known fundamental particle. This coupling is best measured by observing combined production of the Higgs with a top quark pair, a process known as *ttH*. This 'signal' process is predicted to be extremely rare, as it competes with other similar 'background' processes. The most prominent of these is top quark pair production in association with bottom quark pair production, commonly denoted *ttbb+jets*.

In late 2015, the Large Hadron Collider resumed operation at a center-of-mass-energy of 13 TeV. This high-energy, high-luminosity environment brings with it greater chances of observing rare processes such as *ttH*, but produces significantly more background noise. Thus, developing accurate event discrimination algorithms is paramount to the analysis of this elusive process. In recent years, advances in the field of machine learning have produced new computer learning tools, called Deep Convolutional Neural Networks (CNNs). CNNs use the raw input pixels of photographic images to determine for themselves which features best distinguish desired signal events from unwanted background events. By visualizing the data gathered from a high-energy physics detector such as the CMS detector at CERN, we can use CNNs to uncover important information about events.

The goal of this thesis is to investigate the usefulness of applying Deep Convolutional Neural Networks to the search for the *ttH* process. Current results show that CNNs perform as well as standard Artificial Neural Networks in this context, with the potential for future improvements.

**Acknowledgements**

I would like to thank all those who supported me throughout my undergraduate education, including my family, friends, colleagues, professors, and advisors. In addition, I would like to thank the Ohio State University Department of Physics and the Ohio Supercomputer Center for providing me with the resources and funding necessary to complete my work.

**Table Of Contents**

## Chapter 1: Introduction

The Standard Model of particle physics describes many of the fundamental properties and interactions of matter. In order to verify the predictions of the model, numerous scattering experiments are conducted with the Large Hadron Collider (LHC) at CERN. The LHC collides two beams of protons at a center-of-mass energy of 13 TeV, producing showers of stable and unstable particles. By observing these particles with high-resolution detectors along the beam line, aspects of the processes that produced them can be measured.

Information about these processes is gathered using two highly specialized detectors, ALICE (A Large Ion Collider Experiment) and LHCb (Large Hadron Collider beauty), and two general-purpose detectors, ATLAS (A Toroidal LHC ApparatuS) and CMS (Compact Muon Solenoid). However, because many high-energy physics processes can only be observed through examination of their products, it can be difficult to distinguish many important 'signal' processes from the common 'background' processes that produce similar products. The recent upgrades to the LHC provide a higher-energy, higher-luminosity environment that is conducive to the production of numerous rare and interesting physics events. However, the prominence of certain high-energy background processes is much greater in this new environment, necessitating the development of more accurate event discrimination algorithms [1].

One such phenomenon of interest is top quark pair production in association with a Higgs boson, denoted throughout this paper as *ttH*. The top quark is unusually massive compared to all other quarks: the most common quarks, the up and down quarks, are the particles that compose

protons and neutrons, yet the top quark is roughly 185 times more massive than the proton [2]. Fermions gain mass by coupling to the Higgs boson, and the *ttH* process involves a direct coupling of a Higgs boson to a top quark pair. Thus, observation and measurement of this process could provide hints to the nature of the relationship between these two particles.

This process, however, is extremely rare, and is difficult to isolate from its backgrounds. The most prominent background for the *ttH* process is top-quark pair production in association with bottom-quark jets, denoted throughout this paper as *ttbb+jets*. The top quark decays nearly 100% of the time into a bottom quark and a W boson, while the Higgs boson is most likely to decay into a bottom quark pair. The fact that both processes thus produce identical final state particles makes discriminating between the two incredibly challenging [3].

In order to distinguish between highly similar processes such as these, physicists often use computer learning tools called Artificial Neural Networks, or ANNs. An ANN can be thought of as a series of interconnected nodes equipped with activation functions, similar to neurons in a human brain [4]. Though ANNs have provided incredible computational power in past analyses at the LHC, they are somewhat limited in their effectiveness. In the context of high-energy physics, ANNs perform event discrimination based on engineered features, such as invariant mass and angular separation [5]. However, though these variables may provide the most convenient framework in which humans can represent events, they may not be optimal variables for computational discrimination. Indeed, one of the greatest problems facing high-energy physicists today is the fact that ANNs are minimally effective when discriminating between *ttH* and *ttbb+jets*.

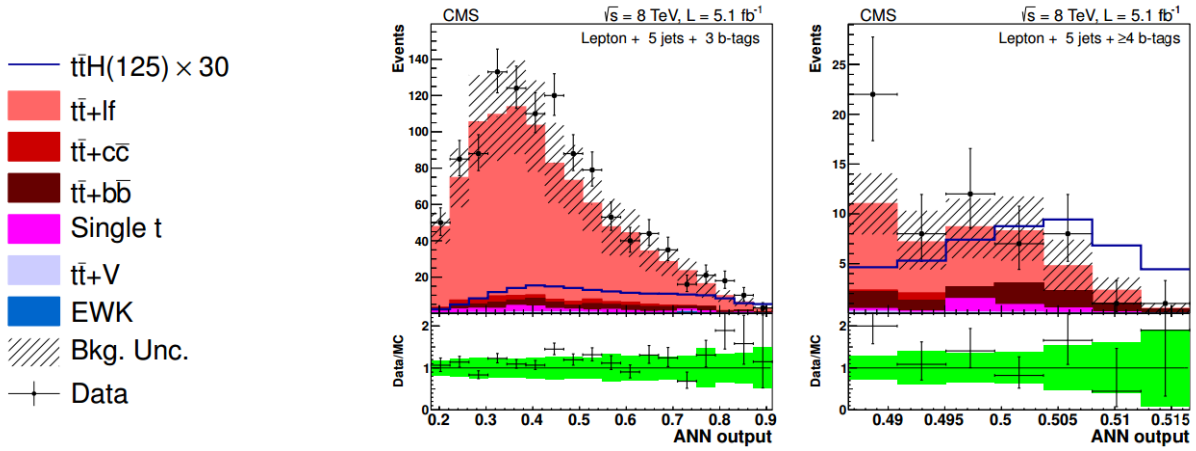



Figure 1: A standard ANN response for ttH and several backgrounds [5].

In **Figure 1**, we see the ANN response variable for *ttH* and its various backgrounds in the analysis performed in [5]. In order to incorporate this network response into an analysis, physicists perform a likelihood fit on the ANN response distribution. However, the similar shapes of the ANN output distributions of *ttH* and *ttbb+jets* (denoted in the Figure as *ttH* and *tt+bb*) suggests that such a method is unlikely to produce optimal results.

Recent advances in the field of machine learning have led to the development of new types of neural networks called Deep Neural Networks [6]. Possessing more layers of nodes than standard Artificial Neural Networks, Deep Neural Networks are able to learn sophisticated features for themselves, rather than rely on engineered features prescribed by humans. Though broader applications of Deep Neural Networks to particle physics are still being explored, one specific application shows promise. Deep Convolutional Neural Networks, or CNNs, are variants of Deep Neural Networks that are specialized for image-processing applications [7]. Rather than learn numerical features of high-energy physics events, Deep Convolutional Neural Networks operate on raw data images gathered from the detectors at the LHC and internalize the geometric

features that characterize the events. In related fields of analysis, CNNs have consistently outperformed other learning algorithms [8]; in preliminary studies, Deep Convolutional Neural Networks perform as well as ANNs in a high-energy physics setting [9]. In this analysis, we investigate the application of Deep Convolutional Neural Networks to the *ttH* classification problem. CNNs provide a fascinating new method of event classification, and hint at a tantalizing solution to this otherwise intractable task.

## Chapter 2: Deep Convolutional Networks

Deep Convolutional Neural Networks are machine learning algorithms optimized for image classification. A CNN will take in a series of labeled two-dimensional images as input, internalize various features of the images, and then utilize these features to classify unlabeled images in accordance with the initial labeling scheme. Though a CNN can be trained to sort images into a number of different categories, a common application is to train the network on 'signal' images, labeled 1, and 'background' images, labeled 0. Each unlabeled image fed into the network during the testing phase will cause the network to produce a value between 0 and 1, which corresponds to how 'signal-like' an image appears.

ANNs and CNNs both belong to a class of neural networks referred to as feed-forward neural networks. Like an ANN, a CNN consists of a number of interconnected stages that function similarly to neurons in the human brain. Each individual neuron in a feed-forward network produces an activation value that is a nonlinear function of the weighted sum of its inputs; this activation value will then become an input value for one or more neurons in the next stage. Common nonlinear activation functions used include the sigmoid and rectified linear unit functions; however, use of the rectified linear unit function has been shown to eliminate many problems that had previously hindered the development of Deep Neural Networks [10]. Many such fully-connected 'hidden' stages are connected to one or more final-state neurons equipped with a Log-Softmax activation function, each of which returns the image's classification value for a given class [11]. This architecture is depicted in **Figure 2.**

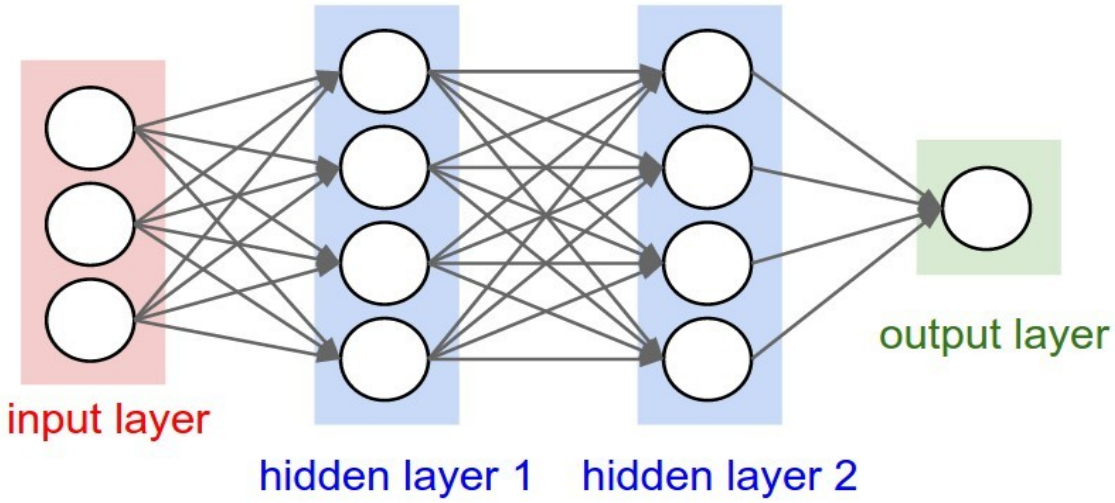



Figure 2: An illustration of an Artificial Neural Network [29].

The error function for a single input image is one-half the squared difference between this calculated label and the actual image label; the average error function is simply the average of all individual error functions for a given set of training images, modulated by a 'weight decay' parameter to prevent overfitting [12].

Initially, the weights in feed-forward networks are randomized [7]. The network 'learns' by iteratively adjusting these weights in order to optimize its classification performance. If E is the average error function, $\nabla E(W)$ is the gradient of this function with respect to the vector of weight parameters, and $\eta$ is a 'learning rate' parameter, then the vector of weights is updated according to the algorithm $W = W - \eta \nabla E(W)$ [13].

This iterative step can be performed after only one input image is passed through the network, or after some fraction of the entire training set is passed. Because it can be computationally intensive to compute the error gradient for the entire training set, it is common practice to break the training set up into batches, each consisting of several input events, and

compute the average error gradient over each batch. GPU acceleration mandates a batch size of 32 events, so our batches consist of 16 signal events and 16 background events [14]. After a single batch is run, the network will update its weight parameters; after all training samples are exhausted, the network will check its performance against an independent test sample. In a typical analysis, the time it takes the network to run over the entire dataset in this manner is referred to as an epoch.

This general framework describes both CNNs and ANNs; the difference between the two lies largely in how input images are encoded and processed. In an ANN, the inputs for each image are the numerical values of engineered features, while a CNN examines raw input images themselves. An image can be thought of as a set of matrices, with each entry corresponding to the intensity in a given pixel. A standard photographic image, for instance, can be represented as three matrices, one containing the intensity of red in each pixel, one containing the intensity of green in each pixel, and one containing the intensity of blue in each pixel. These matrices are represented in the input layer of a CNN. Neurons in the second layer then convolve square receptive fields over the image, producing an activation map for each patch as in **Figure 3**. In the third layer, the dimensionality of the activation maps is reduced by averaging or computing maxima, a process known as spatial pooling. This process is depicted in **Figure 4.** A number of convolutional-pooling layer pairs may follow. The final layers of a CNN are simply a standard ANN; the output of this ANN is the classification probability for a given image. The architecture of a full CNN is depicted in **Figure 5.**

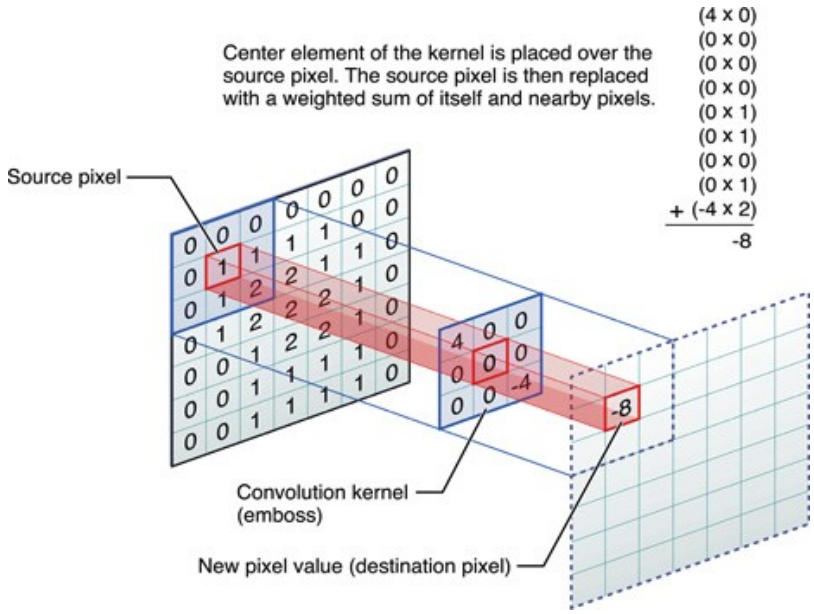



Figure 3: A Convolutional Layer. The value of a given pixel in the feature map is the sum of the source pixels weighted by the convolutional kernel [30].

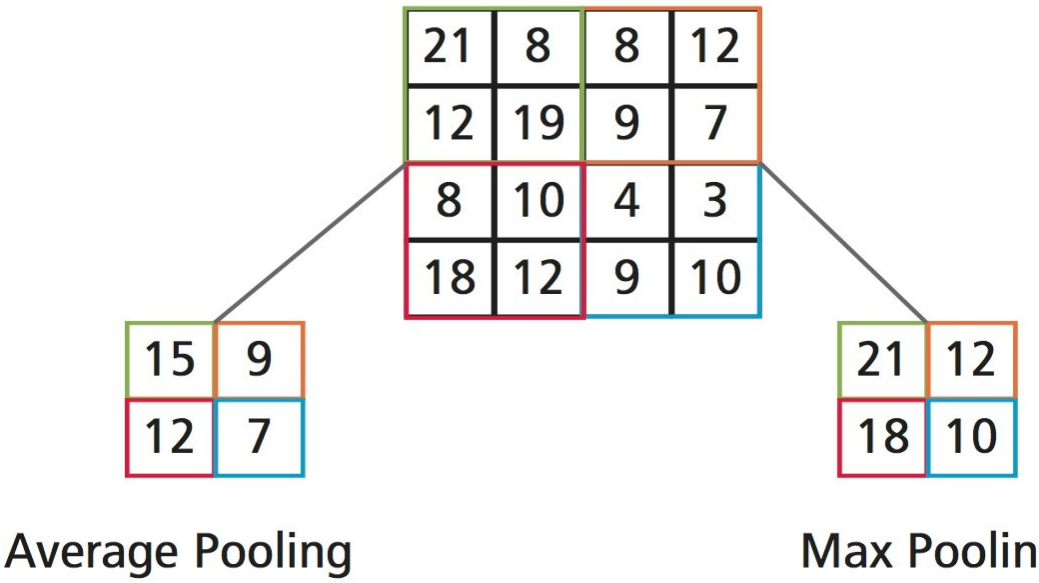



Figure 4: A 2x2 Pooling Layer. On the left, the value of each pooled pixel corresponds to the average value in the similarly-colored large region; on the right, these values correspond to maxima rather than averages [31].

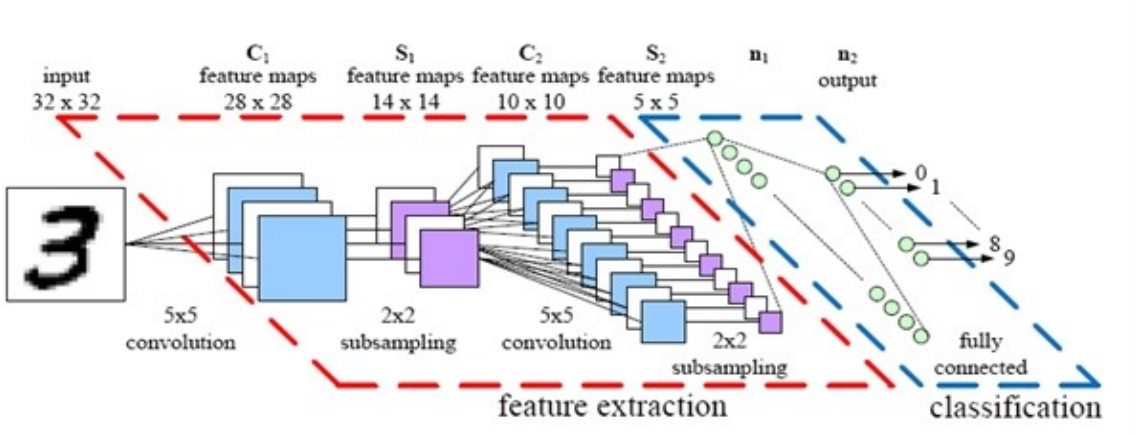



Figure 5: A model of a Deep Convolutional Neural Network. By combining convolution and pooling, the network learns four features at the first stage and 12 at the second. These are then passed to a standard ANN for classification [32].

As the network updates its weights, neurons in each layer learn increasingly sophisticated characteristic features of an image. In a CNN, the adjustment of weights modifies the entire convolutional kernel, rather than individual nodes. This allows the network to search for optimal characteristic features as it scans over the entire image, effectively determining for itself which geometric features can best be used to generalize the content of the image. When a CNN is trained over a dataset containing images of human faces, for instance, low-level features learned include edges and whorls, while high-level features include models of entire faces [15]. Intuitively, then, the more convolutional layers a CNN has, the more complex the features it will be able to learn, and thus the better it will perform. This was confirmed in [7] using photographic images. Similarly, deep ANNs also tend to perform better than shallow ANNs [16]. Measuring the precise benefits of depth is an important current issue in the field of machine learning.

We exploit these tools to perform an analysis of the *ttH* process. In Chapter 3 we discuss the general properties of the *ttH* process and perform an analysis using a traditional ANN. In

subsequent chapters, we discuss how CNNs can be utilized to produce comparable results, and perform an analysis using CNNs.

## Chapter 3: *ttH* Discrimination

As previously stated, the top quark decays with near 100% certainty to a bottom quark and a W-boson. The W boson can then decay either leptonically or hadronically, that is, into a lepton-neutrino pair or a quark-antiquark pair. A Feynman diagram of the process is illustrated in **Figure 6**. In this analysis, we consider a 'lepton+jets' decay mode, in which one of the W-bosons decays hadronically and the other leptonically.

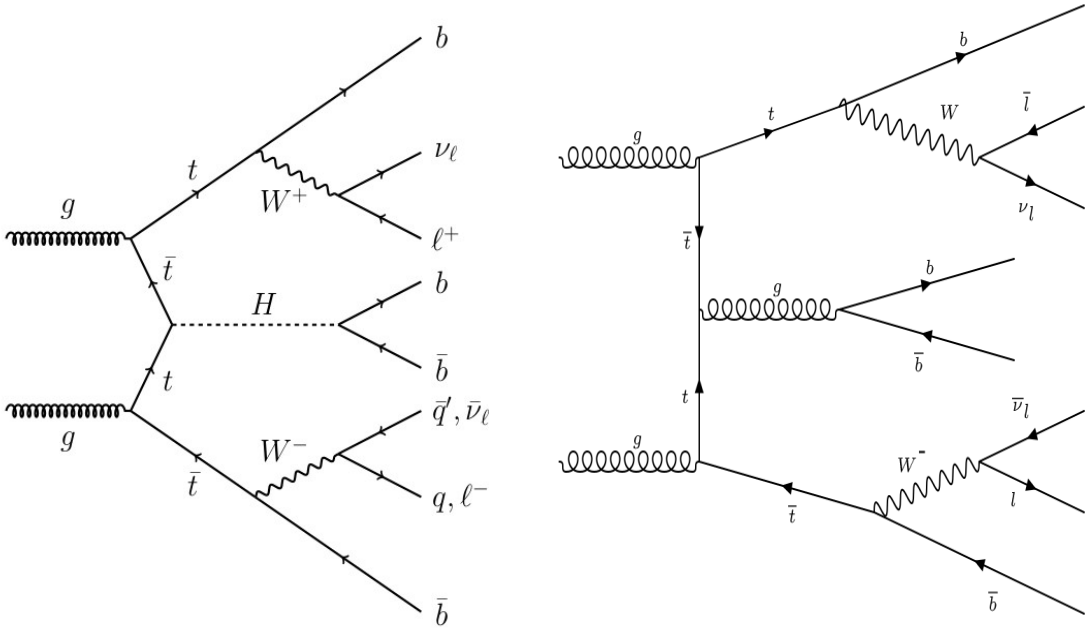



Figure 6: A depiction of a *ttH* event (left) and a *ttbb+jets* event (right) [5].

When quarks are removed from the hadrons they compose, they summon quark-antiquark pairs from the vacuum and bind to them via the strong interaction. This process snowballs, leading to a cascade of hadron production known as a jet. Because free quarks have never been observed in nature, the hadronic behavior of an event must be inferred through the observation

and measurement of these jets [17]. Using the hadronic calorimeter present in a general-purpose detector, we can measure the transverse energies and geometric distributions of these hadronic showers. One of the most useful jet measurement tools is a process known as b-tagging, which allows for determination of the certainty with which a jet can be said to originate from a bottom quark. Because bottom quarks are the heaviest hadron-forming quark, b-tagged jets are typically more energetic than jets originating from other flavors of quark. Furthermore, hadrons containing b-quarks typically travel some distance before decaying, meaning that b-tagged jets appear to originate from a position other than the nominal [18].

Leptons, however, do not interact via the strong interaction, and, with the exception of neutrinos, are much easier to measure. Using the electromagnetic calorimeter and muon spectrometer of a general-purpose detector, we can measure the transverse energy that individual leptons deposit in detector regions. Neutrinos, however, do not interact in a measurable way with the material of the detector: their presence must be inferred by considering conservation laws. The geometry of a general-purpose detector dictates that the sum of all transverse momenta measured should be zero, and, if this sum is nonzero, we can infer that an invisible particle has carried off some portion of the momentum. The momentum carried off by neutrinos in this way is referred to as 'missing transverse energy', or MET.

To measure both leptons and hadrons, general-purpose detectors are equipped with silicon-wafer inner detectors, which provide information on the charges and tracks of particles close to the interaction vertex. These 'tracker' detectors allow for corroboration of information gathered from other detectors, measurement of b-tag probabilities, and rejection of QCD noise[19]. A cut-away slice of the CMS detector is depicted in **Figure 7.**

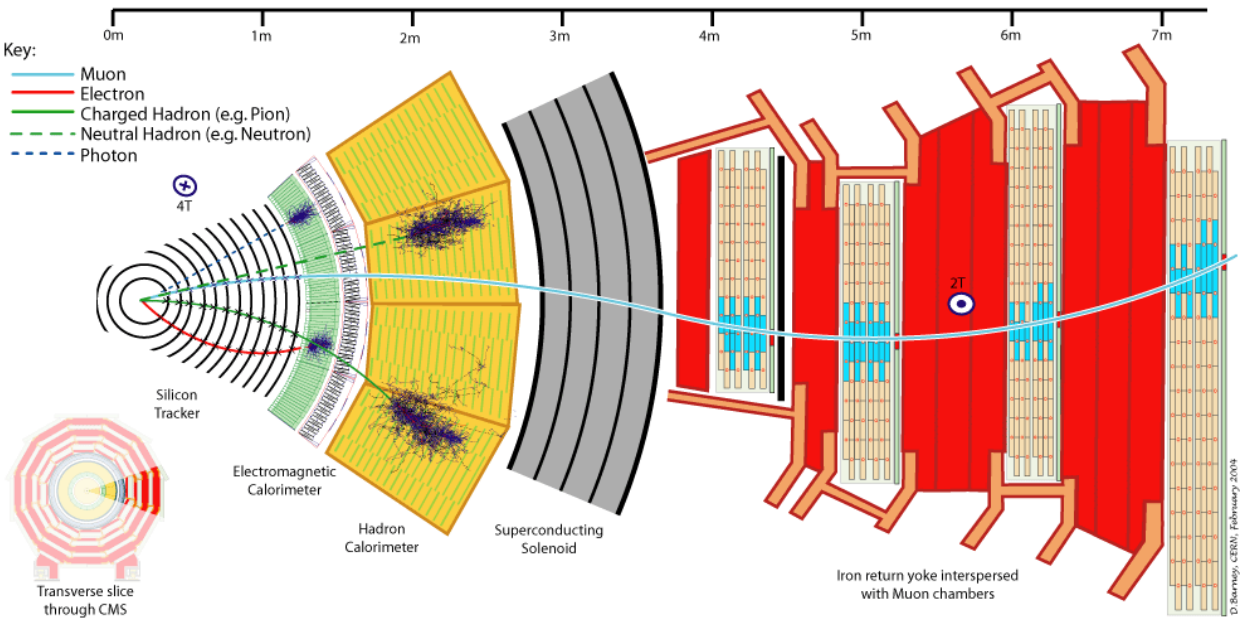



Figure 7: A model of the CMS detector: The detector consists of an inner tracker, two calorimeters, a 3.8T solenoid, and a muon spectrometer. By measuring the amount of energy particles leave in various calorimeters and the curvature of their tracks, the particles' identities can be determined [33].

Previous analyses used tracking data and calorimeter energy measurements to construct features of interest and trained ANNs on these constructed variables. In particular, the CMS analysis in [5] achieved peak performance by directing a four-layer Multi-layer Perceptron ANN to classify events based on variables such as the mean b-tag probability of all jets present in a given event, the average angular separation of all b-tagged jets in an event, and the sum of the lepton, jet, and missing transverse energies in an event. By combining these and other engineered variables, the ANN was able to produce a discriminant variable that was more effective in distinguishing between *ttH* and *ttbb+jets* than any one of the input variables. The ANN performed differently when examining different variables, but it was found that mean b-tag

probability was the most reliable ANN input variable in six of the nine jet multiplicity channels examined [5].

We conduct our own analysis using ANNs in order to obtain a raw classification percentage. Studies regarding the optimization of ANNs are ongoing, particularly regarding the depth of the networks and the nature of the features fed to the network. As such, we examine the results of four networks. One considers only features we deem 'low-level', namely the momentum three-vectors of all objects in the event. Another considers these low-level features as well as the b-tag probabilities of each jet. A third considers the features we deem 'high-level', i.e., the aplanarity and sphericity of the event [20], the reconstructed transverse momenta and pseudorapidities of all leptons and jets, and the missing transverse energy present in the event. The fourth ANN considers these high-level features as well as the b-tag probabilities of each jet.

The low-level networks have the following architecture:

1. n fully-connected neurons, where n corresponds to the number of input variables
2. 4 fully-connected layers of 50 neurons each
3. Log-Softmax Classifier (2 output neurons, one corresponding to each class)

The high-level networks have the following architecture:

1. n fully-connected neurons, where n corresponds to the number of input variables
2. 20 fully-connected neurons
3. Log-Softmax Classifier (2 output neurons)

All input samples are normalized, that is, we subtract the mean of each feature value and divide by the standard deviation. Our background sample includes events containing hadronic jets from sources other than bottom quarks; we refer to this generalized background set as *tt+jets.*

| Level | B-Tags | Best Training Performance | Best Test Performance |
|---|---|---|---|
| Low-level | No | 70.17% | 69.77% |
| Low-level | Yes | 76.50% | 76.19% |
| High-level | No | 69.19% | 68.91% |
| High-level | Yes | 76.34% | 76.22% |

Table 1: ANN performance for various network configurations.

These networks were trained for 18 hours on the Oakley supercomputer cluster at OSC. This provides us with a useful benchmark for traditional network performance. It becomes immediately apparent that low-level features perform, on the average, better than high-level features, suggesting that an investigation of convolutional network performance is well-motivated.

## Chapter 4: A CNN Analysis

In order to properly implement CNNs to perform high-energy physics analyses, we must convert detector data into images. In some sense, the nested sub-detectors of a general-purpose detector can be viewed as high-resolution cameras, gathering information from events and recording it in cylindrical grids. The distributions of energy deposited in different sections of the calorimeter can be unrolled to produce two-dimensional intensity maps similar to the channels of a photographic image. Each sub-detector can be thought of as adding a different channel, analogous to a color channel in a photographic image. Our images contain 5 channels, consisting of the energy distribution in the electromagnetic calorimeter, the energy distribution in the hadronic calorimeter, the energy distribution in the tracker, data from reconstructed leptons, and b-tag probabilities of each jet (recorded at the centroid of each jet). We consider the effects of activating various combinations of these channels.

Particle physicists typically use the variables $\eta$ and $\varphi$ to denote the angular coordinates of phenomena of interest within the detector. $\varphi$ is the azimuthal angle of a given particle track, while $\eta=-\ln\{\tan(\theta/2)\}$, where $\theta$ is the polar angle of a given particle track. See **Figure 8.** We use $\eta$ as our principal coordinate rather than $\theta$ because differences in $\eta$ are Lorentz invariant under longitudinal boosts, and because each unit of $\eta$ contains roughly the same number of particles [21].

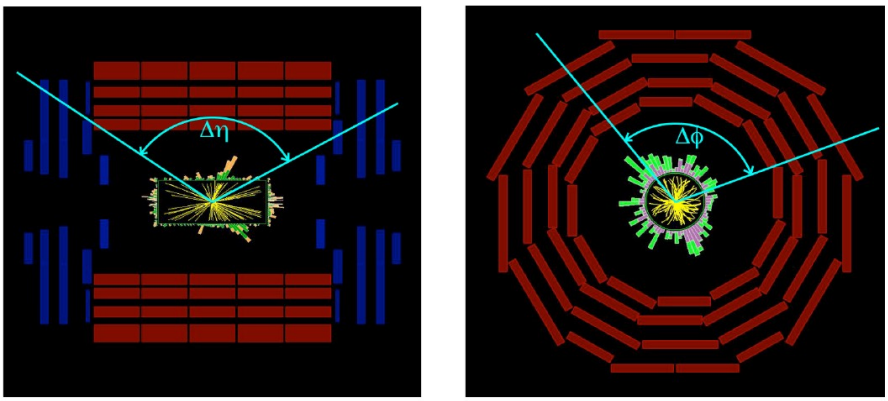



Figure 8: An intuitive depiction of the parameters η and φ, shown here in a general-purpose detector [34].

Using the POWHEG Monte Carlo generator, 799254 *ttH* and 971846 *tt+jets* events are generated at 13 TeV. In *ttH* events, the Higgs is forced to decay to two bottom quarks, but no constraints are imposed on the top quark decays. For the *tt+jets* samples, no constraints are imposed on the top quark decays. For both *ttH* and *tt+jets*, the spacing of proton bunches in the beam is assumed to be 25 ns, and the condition of the LHC beam and detectors (i.e. calibrations, beam spot, and jet energy corrections) are taken as they were when Run 2 data-taking began in Spring 2015 [22]. We use the ParticleFlow event reconstruction algorithm to identify candidate particles from events that are relevant to our analysis [23].

All Particle Flow candidate particles are required to have Pt > 0.2 GeV and $|\eta| < 3.0$. Furthermore, for an event to pass selection criteria, it must have at least one electron or muon and at least five jets. We generate our images in the range $-3.2 \leq \eta \leq 3.2$ and $2\pi \leq \varphi \leq 2\pi$. Our images are 30x30 pixels, that is, 30 η bins by 30 φ bins. We fill five histograms for each event, and use the information in these histograms to define five channels. One channel contains all energy deposited in the electromagnetic calorimeter that is associated with charged leptons (we denote this channel Lepton), one contains all energy deposited in the electromagnetic calorimeter

that is associated with neutral particles (Neutral EM Energy), one contains all energy deposited in the hadronic calorimeter that is associated with neutral particles (Neutral Hadronic Energy), and one contains all energy deposited in the hadronic calorimeter that is associated with charged particles (Charged Track Energy). By examining these channels, we are able to accurately model the energy depositions left in the detector by a high-energy physics process. We also add a fifth channel, the b-tag probabilities of each jet (recorded at the centroid of each jet), using the ParticleFlow b-tag-probability information. Sample images are depicted in **Figure 9** and **Figure 10**. We do not include B-tag information on the plots, since it does not have the same units as the information in the other channels.

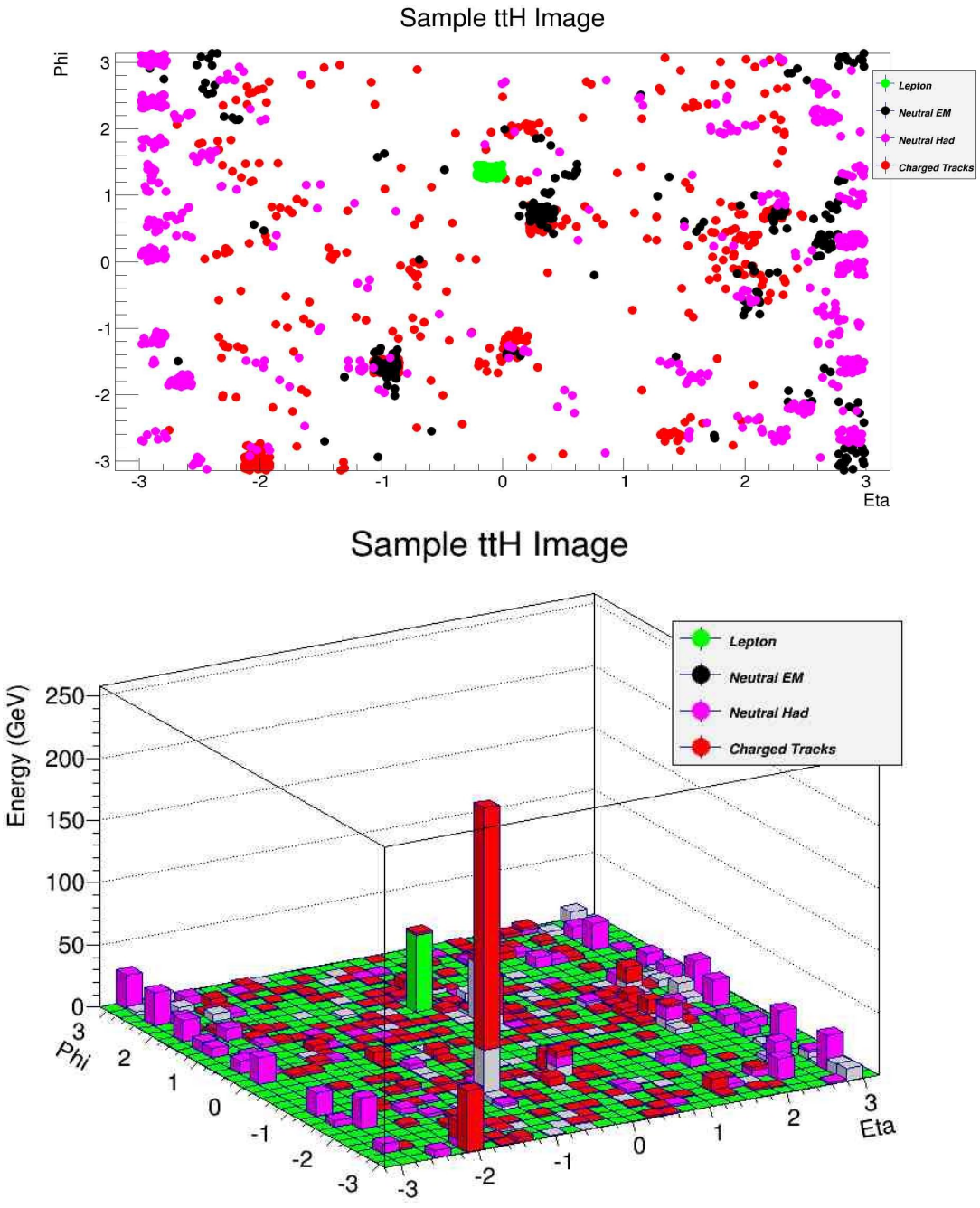



Figure 9: A *ttH* event image.

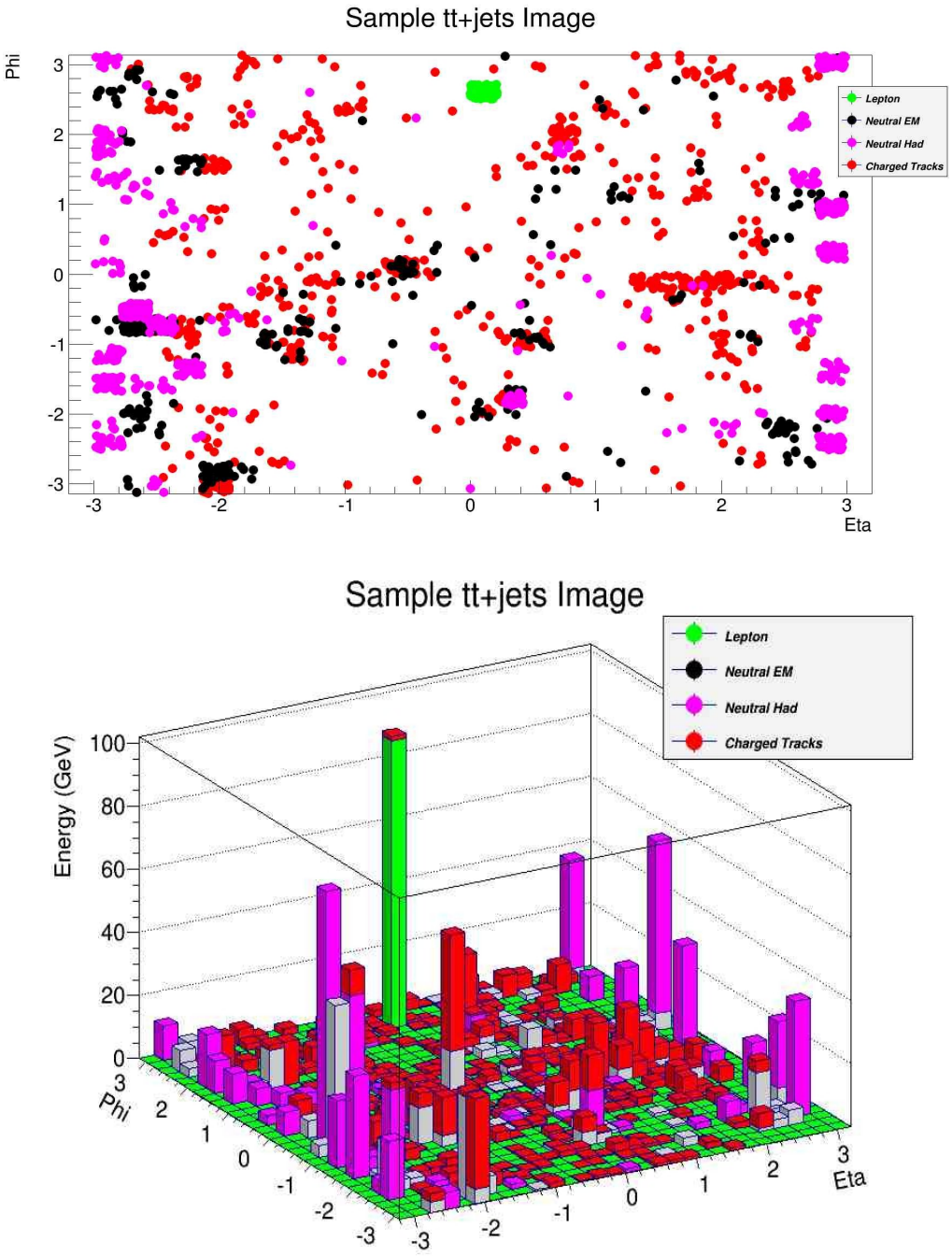



Figure 10: A *tt+jets* image.

After generating these images, we train a series of CNNs to analyze them. We design these networks with Torch7, a free software framework designed for Graphics Processing Unit (GPU) accelerated machine learning [14]. GPUs are specialized processors optimized for high-speed image-processing applications; utilizing them drastically decreases the time it takes to train a CNN. We train our networks using the computational resources of the Oakley Cluster at the Ohio Supercomputer Center. Each network is trained for 18 hours: this allows enough time to reach roughly asymptotic performance while still permitting regular checks.

Prior to training, we break the signal and background datasets up into 54 chunks, each containing 14801 events (more *tt+jets* events were generated, but we ignore these in order to construct a training set consisting of equal proportions signal and background). We set aside one chunk each from signal and background and merge them to create an independent test set. We then select one signal chunk and one background chunk from the remaining files and merge them to form an initial training set, train and test the network on this dataset, then select a new (unused) signal and background chunk pair and repeat the process. After all training chunks have been depleted, we repeat the full training set. This segmentation procedure allows us to train our network over large datasets while avoiding memory constraints, and does not adversely affect network performance. We define an epoch as the time it takes the network to run over one of these training chunks, rather than over the entire dataset.

In order to train our networks, we first combine the signal and background data, shuffling the events together randomly. This prevents the network from favoring one particular class of data while still maintaining equal proportions of the two in the dataset. Next, we repeatedly propagate events through the network in batches of 32, updating neuron weights after each batch.

After the training chunk has been exhausted, we record the network's classification performance. We then record the network's performance on the independent test set. In many cases, networks can 'overtrain', learning features that are exclusive to the training set; this test set is therefore necessary for evaluating a CNN's true performance [24].

We constructed our initial model based on a 'Face Detector' convolutional network defined in [25]. This model was optimized for the detection of complex features in human faces, and, as such, was a useful starting point for our network. After sufficient modification of network parameters, our model was as follows:

1. Input n-channel 30x30 image, where $1 \leq n \leq 5$
2. Convolutional layer with 10 3x3 features
3. 2x2 Spatial Max Pooling layer
4. Convolutional layer with 5 5x5 features
5. 125 fully-connected neurons
6. 10 fully-connected neurons
7. Log-Softmax classifier (2 output neurons, one corresponding to each class)

Network weights are initialized randomly by Torch7 [14]. We use the rectified linear unit as our activation function, for reasons outlined in previous chapters. We explore the applications of a number of data processing techniques, including channel selection, normalization, and channel combination, in order to determine their effects on network performance.

## Normalization

We apply a normalization scheme to the data prior to training. In previous studies involving CNN responses to photographic image data, normalization of inputs appeared to produce best results [7]. We investigate the applicability of this claim to our physics data. For every input image, we loop over each channel, calculating the mean and standard deviation of pixel intensity for each channel. We then subtract the mean from each pixel intensity value and divide the result by the standard deviation. We train two four-channel networks and two five-channel networks (i.e., with and without b-tags), one of each with the normalization scheme implemented, and one of each without.

## Channel Selection

We 'toggle' various channels on and off, and investigate the network's performance for each channel combination. In particular, we pay special attention to the network's performance with and without the B-tag channel, as well as the network's performance exclusively on the B-tag channel. This gives us a rough information of the information content in each channel. All channels are normalized for this process.

## Channel Combination

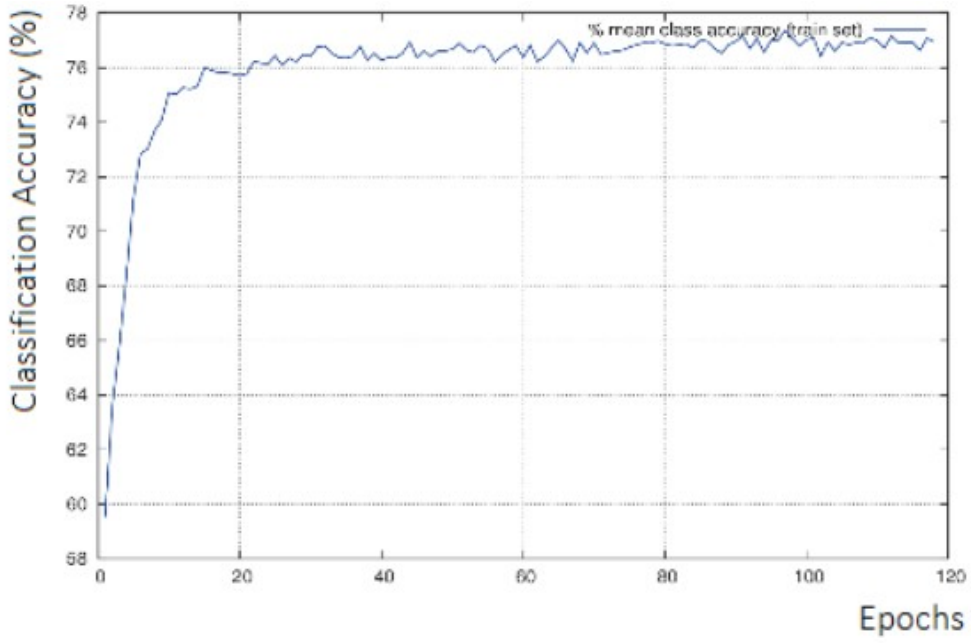
We normalize the data and add the energy channels together (that is, all channels but the b-tag information channel). We compare the network performance with and without b-tag information using this constructed channel. We thus train two networks at this stage, one over two-channel images (total energy and b-tags) and one over one-channel images (total energy

only). This helps us determine whether or not the nature of the energy deposit is relevant for image classification.

By processing the data in these ways, we are able to develop insight into how the network learns to classify images. In future studies, we hope to optimize the network architecture for this analysis, perhaps by implementing techniques such as Dropout, which masks certain fully-connected neurons in order to prevent overtraining [24], and by altering the learning rate, a parameter which determines how the network adjusts its weights. Most importantly, we hope to optimize feature size to better correspond to estimated jet sizes and separations. We describe some proposed architectures in the **Appendix**, and discuss a procedure by which to find allowed kernel sizes for these architectures.

## Chapter 5: Results

A sample CNN performance plot is shown in **Figure 11**. This network is trained on all 5 channels, each of which is normalized prior to input. We can see that the network is able to learn to effectively distinguish between the signal and the background after only a few epochs, and that classification accuracy quickly reaches a plateau. The network performance does appear to increase slightly over time after reaching the plateau phase, but gains decrease rapidly. This indicates that the networks may benefit from longer training times; we suggest taking this into account in future studies. We note that the training performance is a fraction of a percent better at plateau than the test performance.



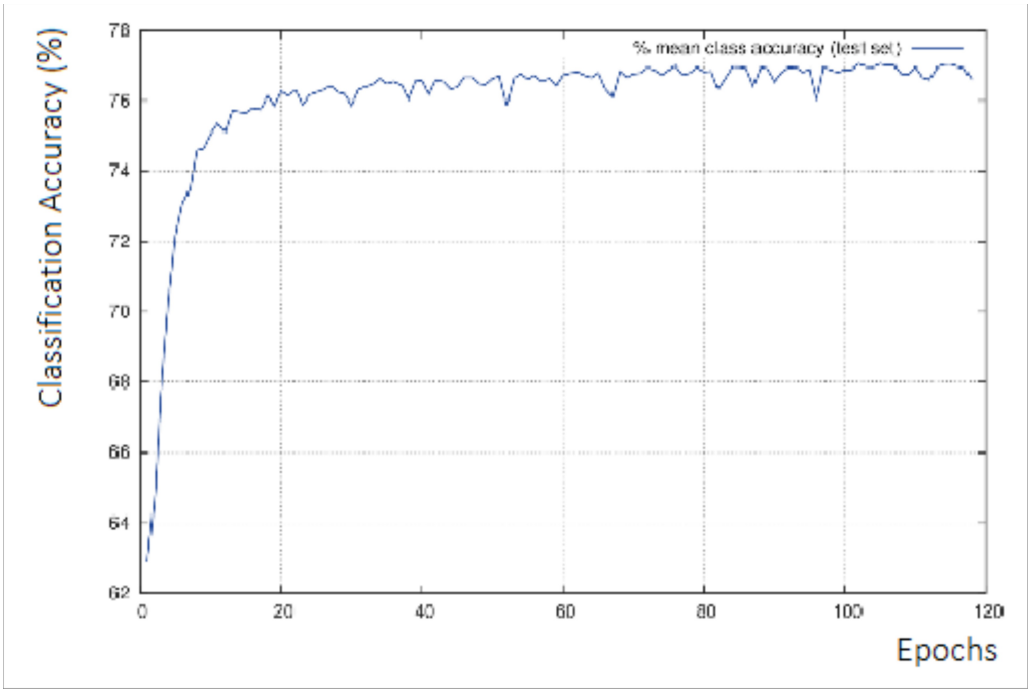



Figure 11: Network performance over time. The image on the preceding page depicts training accuracy, while the image above depicts test accuracy. We note that, though performance mostly appears to have reached a plateau, there do still appear to be some incremental gains.

Varying the random number seed of the weight initialization corresponds to fluctuations in accuracy of roughly ±0.3%; we use this statistic as an estimate of the uncertainty on all measurements [9].

## Normalization

We train four networks, varying whether or not the b-tag channel is included and whether or not the channels are normalized. The results are summarized in **Table 2**.

| Normalized | B-Tags | Best Training Performance | Best Test Performance |
|---|---|---|---|
| No | No | 68.05% | 67.91% |
| No | Yes | 68.06% | 67.97% |
| Yes | No | 68.35% | 68.19% |
| Yes | Yes | 77.17% | 77.05% |

Table 2: CNN performance for various normalization settings.

We conclude that, in general, normalization does appear to significantly increase classification accuracy. When b-tags were not included, normalization produced minimal gains, but when tags were included, normalization led to dramatic increases in performance. Furthermore, including b-tagging information when the data is not normalized had almost no effect on classification accuracy. This suggests that, due to the different scale of the information in the b-tag channel, b-tag data must be normalized in order for the network to utilize it. Even beyond this, though normalization does not have as significant of an effect on the data when tags are not involved, we determine that it is still a worthwhile procedure. Unless explicitly stated otherwise, we henceforth normalize all data before training.

## A Comparison to ANN Analysis

We can compare **Table 1** and **Table 2** to gain an understanding of CNN performance relative to traditional ANN performance. The normalized, b-tag-inclusive CNN performed the best out of all networks, roughly 0.9% better than the next-best network. At the current time, it appears that, if b-tag information is not provided, low-level ANNs perform the best. However, CNNs clearly perform comparably. We suggest that, when properly optimized, CNNs may have

the potential to outperform ANNs even when b-tag information is not included.

Having verified the impressive potential of b-tagging information, we determine how important the information in the other channels is to network performance. We do this in two ways, first by combining the energy channels to remove information about the specific nature of each energy deposit, then by removing various channels.

## Channel Combination

Before loading in data from each event, we combine the energy channels. First, we normalize all channels, including b-tag information. Next, we create a new channel, systematically defining pixel intensity in this new channel to be the sum of the intensities for a given pixel in all other energy channels. We train two CNNs as defined in the preceding chapter. We record the results in **Table 3**.

| B-Tags | Best Training Performance | Best Test Performance |
|---|---|---|
| No | 68.22% | 67.88% |
| Yes | 77.35% | 77.32% |

Table 3: CNN performance on merged data.

Surprisingly, without the b-tag channel, the network appears to perform only slightly worse when the energy channels are added together, while with the b-tag channel, the network actually appears to improve its performance slightly. This suggests that the content of each individual channel may not play as significant of a role in determining classification accuracy as a combination of all channels.

## Channel Selection

We normalize each channel and toggle them on and off. We train CNNs on all possible combinations of the five channels. Though we cannot train a network on no input, in the row corresponding to zero input channels, we report an accuracy of 50%, as this is the classification accuracy corresponding to random guessing. Likewise, the options to examine all channels and to examine all channels but b-tag probability were discussed previously; the results from the previous sections were used to fill in the table. We thus need only train 29 CNNs in order to cover all channel combinations. We report the results in **Table 4.**

| Lepton | Neutral EM | Neutral Hadronic | Charged Track | B-tag Probability | Best Train Performance | Best Test Performance |
|---|---|---|---|---|---|---|
| Y | Y | Y | Y | Y | 77.17% | 77.05% |
| Y | Y | Y | Y | N | 68.35% | 68.19% |
| Y | Y | Y | N | Y | 77.38% | 77.05% |
| Y | Y | Y | N | N | 66.92% | 66.25% |
| Y | Y | N | Y | Y | 77.54% | 77.28% |
| Y | Y | N | Y | N | 68.28% | 67.19% |
| Y | Y | N | N | Y | 77.43% | 77.25% |
| Y | Y | N | N | N | 66.50% | 66.09% |
| Y | N | Y | Y | Y | 77.43% | 77.01% |
| Y | N | Y | Y | N | 67.04% | 66.84% |
| Y | N | Y | N | Y | 77.15% | 76.66% |
| Y | N | Y | N | N | 60.07% | 59.91% |
| Y | N | N | Y | Y | 77.23% | 77.05% |
| Y | N | N | Y | N | 67.08% | 67.20% |
| Y | N | N | N | Y | 77.00% | 76.62% |
| Y | N | N | N | N | 54.55% | 54.22% |
| N | Y | Y | Y | Y | 77.48% | 77.14% |
| N | Y | Y | Y | N | 68.31% | 68.17% |
| N | Y | Y | N | Y | 77.44% | 77.19% |
| N | Y | Y | N | N | 66.47% | 65.91% |
| N | Y | N | Y | Y | 77.50% | 77.44% |
| N | Y | N | Y | N | 68.30% | 67.99% |
| N | Y | N | N | Y | 77.48% | 77.27% |
| N | Y | N | N | N | 66.39% | 66.06% |
| N | N | Y | Y | Y | 77.25% | 77.10% |
| N | N | Y | Y | N | 67.42% | 67.21% |
| N | N | Y | N | Y | 77.09% | 76.70% |
| N | N | Y | N | N | 59.17% | 59.36% |
| N | N | N | Y | Y | 77.35% | 77.06% |
| N | N | N | Y | N | 67.13% | 67.20% |
| N | N | N | N | Y | 76.89% | 76.63% |
| N | N | N | N | N | 50.00% | 50.00% |

Table 4: CNN performance on all possible channel combinations. All data is normalized prior to training.

We note a few interesting facts about the data:

1. Training on the B-tag probability alone led to more than 76% classification accuracy. This suggests that the information provided by this channel 'overrides' the other information; the network appears to rely most heavily on it to classify events.
2. The Charged Track and Neutral EM channels appear to be more information-dense, compared to other non-B-tag channels. When trained on these channels alone, the network performs roughly as well as it does when these two channels are allowed in combination with others. However, the information the network learns from these channels may be repeated: the network performs roughly as well trained on these channels together as it does on each of them individually.
3. Similarly, the Lepton and Neutral Hadronic channels appear to be less information-dense: when trained on these channels alone, the network performs relatively poorly. When trained on these channels together, the network also performs relatively poorly.

Upon examining the number of jets in an event with b-tag probability greater than 0.8, it was found that *ttH* events most commonly had three b-tagged jets, while *tt+jets* events most commonly had two. This suggests that the network was able to classify images relatively accurately simply by counting the number of b-tagged jets in an event. In order to remedy this, we implement a cut on the number of b-tagged jets in an event prior to selection, mandating that the number of tags in each event be comparable. This forces the network to learn to discriminate based on the geometry of the b-tagged jets rather than their multiplicity, and isolates the

*ttbb+jets* background from the more general *tt+jets* sample. When we require that events contain exactly three b-tagged jets, the dataset is reduced to 309036 *ttH* events and 35658 *tt+jets* events.

When trained on this reduced dataset, the network is able to obtain a training classification accuracy of 69.773% and a testing classification accuracy of 68.935%. This suggests that, when fed comparable sets of signal and background images, CNNs are able to successfully discriminate between *ttH* and *ttbb+jets*.

# Chapter 6: Proposed Future Studies

As previously mentioned, the most immediate extension of these results is an optimization of network architecture. In particular, we can explore the relative performance of the models mentioned in the **Appendix**, as well as the use of techniques such as Dropout and alteration of the learning rate. We can also modify the depth of each network, as network depth has in many cases been shown to dramatically increase performance.

Because the data images are derived from previously-generated files, it is relatively easy to change the resolution of the images. When the input images contain more pixels, classification accuracy may increase, as the amount of visible structure in learned features will likewise increase. We propose to test the effects of image resolution on network performance by representing the dataset as images of other sizes, such as 90x90 and 10x10, and feeding this data into an adapted version of the current CNN model.

## Scene Labeling

Another future application, called scene labeling, may allow for improvement upon analyses of jet topologies in calorimeter images. In a scene labeling procedure, each individual pixel in an image is tagged as belonging to one of several predetermined categories. To produce these tags, a CNN is modified to learn specific features that are correlated with labeled regions in an image [26]. A sample CNN scene-labeling algorithm, trained to identify various regions of an landscape scene, is depicted in **Figure 12.**

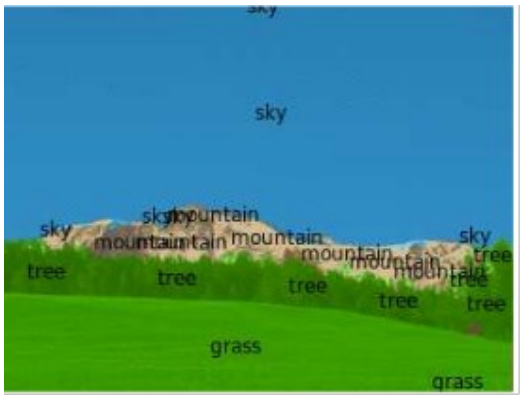



Figure 12: A scene-labelled image. Each pixel is marked to correspond with the region or object it represents [26].

By using scene labeling to mark the regions of high-energy physics images that contain certain types of jet, such as heavy quark jets and gluon jets, it may be possible to utilize CNNs to recover valuable information about the topology of rare events. This may have potential applications in the development of new b-tagging algorithms, as well as the development of tagging algorithms for other, less distinctive types of jet.

## Trigger Applications

During its current run, it is estimated that the CMS detector will gather roughly 40 terabytes of data every second, roughly corresponding to 40 million events per second [19]. Most of this information comes from relatively well-understood events, such as the production of low-energy jets. Due to the inability to store all of this information, a series of high-speed triggers are designed to prune data before storage, discarding events that are deemed uninteresting for physics purposes. Example event selection criteria include the presence of an electron or muon, a large missing transverse energy, or the presence of a high-energy jet. It is entirely possible that a

CNN could learn to recognize the geometries of events of interest and function as an independent triggering algorithm, or, if scene labeling is implemented, could aid an existing trigger algorithm in picking out triggering characteristics.

Current trigger systems rely largely on field-programmable gate arrays, or FPGAs, complex circuits whose structures can be altered dynamically. Studies involving the implementation of Deep Convolutional Neural Networks on FPGA systems have proved promising [27], so it may be interesting to investigate the applicability of these results to the high-energy LHC environment.

## Conclusion

We have shown that a Deep Convolutional Neural Network trained on raw calorimeter data is able to discriminate between simulated 13 TeV *ttH* and *ttbb+jets* events as well as or better than a traditional ANN. This result suggests that, with further optimization, CNNs may be the key to obtaining a measurement of the top-Higgs coupling.

## Appendix

We present a useful formula for determining allowed sizes of convolutional kernels and the output at each stage.

Assume the input of a convolutional layer is a set of n by n images with m channels. Let the first layer be q neurons deep, with kernel size k by k. We see that, as the convolutional layer takes steps over the image, the layer will produce q feature maps with dimension (n-k+1) by (n-k+1) . If this layer is immediately followed by a pooling layer with kernel size p by p, we see that the output is q feature maps of size $\frac{(n-k+1)}{p}$ by $\frac{(n-k+1)}{p}$ . These feature maps will then become the inputs of the next layer.

We see that, at each stage, the feature map sizes must be integers. We are thus constrained in our choices of kernel size by the criterion p|(n-k+1) for each later. As we add convolutional layers, we are forced to consider this constraint for several layers of neurons.

Our input images are 30 by 30. Given the constraint (and the fact that we disregard kernels of size 1 and 30), the allowed first-layer convolutional kernels are of sizes 3,4,5,6,7,9,10,11,13,15,16,19,21,22,23,25, and 27. We note that the average jet has a radius of roughly 1.91 pixels in our 30 by 30 image [28] while the average jet separation varies with the number of b-tags [5], so we propose some alternate architectures to study network performance at this scale. Many other architectures are possible, but we design models to maximize the image size after the first convolutional-pooling layer in order to take advantage of later layers.

We list these models in **Table 5.**

| Name | K1 | P1 | Layer 2 Input | K2 | P2 | Layer 2 Output |
|---|---|---|---|---|---|---|
| Model 1 | 3 | 2 | 14 | 5 | 2 | 5 |
| Model 2 | 3 | 2 | 14 | 3 | 2 | 6 |
| Model 3 | 7 | 2 | 12 | 5 | 2 | 4 |
| Model 4 | 7 | 2 | 12 | 3 | 2 | 5 |
| Model 5 | 11 | 2 | 10 | 5 | 2 | 3 |
| Model 6 | 11 | 2 | 10 | 3 | 2 | 4 |

Table 5: Some proposed network architectures. These satisfy the kernel size constraint and allow us to better study features such as jet size and separation.

In future network optimization studies, we hope to explore the performance of these architectures relative to the current model.

## Works Cited


[1] Chatrchyan, Serguei, et al. . "Projected performance of an upgraded CMS detector at the LHC and HL-LHC: contribution to the Snowmass process." *arXiv preprint arXiv:1307.7135* (2013).

[2] Aad, Georges, et al. "First combination of Tevatron and LHC measurements of the top-quark mass." *arXiv preprint arXiv:1403.4427* (2014).

[3] K.A. Olive *et al.* (Particle Data Group), Chin. Phys. C, 38, 090001 (2014) and 2015 update.

[4] Rosenblatt, F. Psychological Review, Vol 65(6), Nov 1958, 386-408. http://dx.doi.org/10.1037/h0042519. The Perceptron: a probabilistic model for information storage and organization in the Brain.

[5] Chatrchyan, Serguei, et al. "Search for the standard model Higgs boson produced in association with a top-quark pair in pp collisions at the LHC." *Journal of High Energy Physics* 2013.5 (2013): 1-47.

[6] Hinton, Geoffrey E. "Learning multiple layers of representation," *Trends in Cognitive Sciences*, 11, pp. 428–434, 2007.

[7] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

[8] Dieleman, Sander, Kyle W. Willett, and Joni Dambre. "Rotation-invariant convolutional neural networks for galaxy morphology prediction." *Monthly Notices of the Royal Astronomical Society* 450.2 (2015): 1441-1459.

[9] Timcheck, Jonathan P. "Image Classification Applied to High Energy Physics Events." (2015).

[10] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." *International Conference on Artificial Intelligence and Statistics*. 2011.

[11] Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. Neurocomputing (pp. 227-236). Springer Berlin Heidelberg

[12] Moody, J., et al. "A simple weight decay can improve generalization." *Advances in neural information processing systems* 4 (1995): 950-957.

[13] LeCun, Yann A., et al. "Efficient backprop." *Neural networks: Tricks of the trade*. Springer Berlin Heidelberg, 2012. 9-48.

[14] Collobert, Ronan, Koray Kavukcuoglu, and Clément Farabet. "Torch7: A matlab-like environment for machine learning." *BigLearn, NIPS Workshop*. No. EPFL-CONF-192376. 2011.

[15] Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009.

[16] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation* 18.7 (2006): 1527-1554.

[17] Ali, Ahmed, and Gustav Kramer. "Jets and QCD: A historical review of the discovery of the quark and gluon jets and its impact on QCD." *The European Physical Journal H* 36.2 (2011): 245-326.

[18] Chatrchyan, Serguei, et al. "Identification of b-quark jets with the CMS experiment." *Journal of Instrumentation* 8.04 (2013): P04013.

[19] Chatrchyan, Serguei, et al. "The CMS experiment at the CERN LHC." *Jinst* 3.08 (2008): S08004.

[20] J.D. Bjorken and S.J. Brodsky, Statistical model for electron-positron annihilation into hadrons, Phys. Rev. D 1 (1970) 1416 [ IN SPIRE ].

[21] Hama, Yogiro. "A note on Lorentz transformation and pseudo-rapidity distributions." *Journal of the Physical Society of Japan* 50.1 (1981): 21-23.

[22] Alioli, Simone, et al. "A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX." *Journal of High Energy Physics* 2010.6 (2010): 1-58.

[23] CMS collaboration. "Commissioning of the particle-flow event reconstruction with the first LHC collisions recorded in the CMS detector." *CMS Physics Analysis Summary CMS-PAS-PFT-10-001* 30 (2010).

[24] Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

[25] Farabet, Clement, et al., Demos & Turorials for Torch7, (2014). https://github.com/torch/demos/

[26] Farabet, Clement, et al. "Learning hierarchical features for scene labeling." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.8 (2013): 1915-1929.

[27] Farabet, Clément, et al. "Cnp: An fpga-based processor for convolutional networks." *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*. IEEE, 2009.

[28] Huth, John E., et al. "Toward a standardization of jet definitions." *Presented at*. No. FERMILAB-CONF-90-249-E. 1990.

[29] Karpathy, Andrej. “CS231n: Convolutional Neural Networks for Visual Recognition”, (2015). Stanford University. (http://cs231n.github.io/neural-networks-1/).

[30] Apple, Inc. “vImage Programming Guide” (2011). Mac Developer Library. (https://developer.apple.com/library/mac/documentation/Performance/Conceptual/vImage/Introduction/Introduction.html)

[31] Hijazi, Samer, Rishi Kumar, and Chris Rowen. "Using Convolutional Neural Networks for Image Recognition", (2015). Cadence Design Systems.

[32] Andreev, Sergey, et. al. “Efficient mapping of the training of Convolutional Neural Networks to a CUDA-based cluster” (2016). Parallel Architecture Research Eindhoven. (http://parse.ele.tue.nl/)

[33] Barney, David. "CMS Detector Slice" (Jan 2016). CMS-PHO-GEN-2016-001", (https://cds.cern.ch/record/2120661). CMS Collection.

[34] Pivarski, Jim. “My last LHC status update” (Sept. 2010). The Everything Seminar. (https://cornellmath.wordpress.com/)